Appendix A

ts:view

```
<ts:view name="namechars" scrolling="yes|no" width="pixels | percentage" height="pixels | percentage" txnPriority="int"
txnBandwidth="int" txnResolution="server" txnView="namechars" font="normal|bold" fg="#000000"
moveFocus="none|first">...</ts:view>
```

Defines a screen or the content of a ts:popupMenu, ts:popupView, or html:frame. Provides scope for rule matching. An application must have a view named 'main', which is the initial view. Must contain html:body unless this view is used by a ts:popupMenu or this view defines a frameset.

Attributes

Attribute	Description
name	The name of this view. Required.
scrolling	Whether to show a scrollbar in the view. Optional. Default is yes. Valid values: yes, no. no prevents a scrollbar. yes adds a scrollbar, if the view contains any items.
width	The width of the view, in pixels or percentage. Applies to views used by ts:popupView; otherwise ignored.
height	The height of the view, in pixels or percentage. Applies to views used by ts:popupView; otherwise ignored.
txnPriority	Controls which data is sent during synchronization. Used by the ts:sync element. Valid values: integers 1-5, with 1 being the highest priority. The default is 3.
txnBandwidth	Controls which data is sent during synchronization. Used by the ts:sync element. Valid values: floating point values from 0.0 to large numbers (kilobits/s), -1, or "detect". The default value is -1 (meaning no bandwidth specification). The value "detect" determines the current connection speed and only sends transactions which have overall bandwidth values less than or equal to the current connection speed.
txnResolution	Controls whether the server's edit value or the client's edit value is used when the server and client report different edit values. Used by the ts:sync element. "server" indicates that the server value prevails when resolving the edit conflict. "client" indicates that the edit performed on the client prevails. The default is "server".
txnView	Specifies what view to show if there is a conflict while synchronizing. Allows for the application author to decide how the conflict viewing should look for edit conflicts occurring in a particular view. If no conflict occur, nothing special happens. But if a conflict occurs, the client goes into action. It looks for a view with a reserved name, conflictViewer. If none is found, it displays a dialog box saying that conflicts occurred. If set to "first", it displays the specified view. Optional. See also Specifying How Conflicts Appear when Synchronizing.
font	The style of the text in the view. Valid values: bold, normal. The default is normal.
fg	The foreground color of the view, in HTML RGB format, such as fg='#00FF00' for green. For more information, see Using Color in User-interface Elements.
moveFocus	Whether to put an initial edit focus somewhere in this view. Valid values: none, first. Default is "none". If set to "none", the view will not put initial focus on any editable control that is contained in the view. If set to "first", the view will put initial focus on a control. The control that receives focus will be the first control that has a focalPoint attribute equal to "yes". See Setting the Focus on a UI Element.

Children

For regular views (screens): html:body, html:frameset; ts:menuitem, ts:deleteTxn, ts:demandPage; xsl:variable, (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

For views used by popupMenu: ts:field, ts:link; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained In

ts:servo.

ts:expandable

`<ts:expandable expanded="false|true" empty="{true}" font="normal|bold" fg="#000000"`
`moveFocus=none|first">...</ts:expandable>`

A user-interface control that expands when clicked. Initially appears as a solid right-pointing triangle and optional visible content.

When the initially visible content or the arrow is clicked, the control expands to also show hidden content. The arrow becomes an empty triangle.

Attributes

Attribute	Description
expanded	Whether to initially display the control in the expanded state. Optional. Valid values: true, false. The default is false.
empty	Indicates whether the arrow of the expandable should be gray. Optional. Valid values: true, false. The default is false.
	This affects the expandable's appearance, not its behavior. If the expandable contains content and the user expands it, the content will still appear even though the arrow is gray. You can include tagging to automatically evaluate whether the expandable is empty and set this attribute (note that this can affect performance).
	The value is an attribute value template, so it can be turned on or off based on a value in a data tree; you can specify "true" or "false", or write an expression that evaluates to true or false in curly braces, such as "{$isEmpty}". See http://www.w3.org/TR/xslt#attribute-value-templates
font	The style of the text in the expandable control. Valid values: bold, normal. The default is normal.
	If unspecified, inherited from closest ancestor with font attribute.
fg	The foreground color of the button, in HTML RGB format, such as '#00FF00' for green. For more information, see Using Color in User-interface Elements
moveFocus	Whether to consider putting the initial edit focus and cursor on this expandable when the user opens the view that contains this expandable. In the view, the first expandable that has moveFocus="first" will get focus. Valid values: none, first. Default is "none". If set to "none", the expandable will be considered for initial focus. If set to "first", focus is set to this expandable if this is the first expandable in the view. See Setting the Focus on a UI Element.

Children

ts:visible_ts:hideable, ts:onClick_ts:deleteTxn; xsl:variable; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained in

ts:hideable; html:body, html:td (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

Remarks

This element contains a UI object which becomes the header for the control, followed by another UI object, which becomes the hidden content. Any UI object item can appear inside of an expandable control. For example, fields and inline tables can be in expandable object t.

The ts:visible subelement contains the information to display when collapsed. If this is specified, the information is shown and then the arrow is placed to the right. If none is specified, only the arrow appears.

The ts:hideable subelement contains the information to display when expanded.

ts:hideable

`<ts:hideable font="normal|bold" fg="#000000">...</ts:hideable>`

The additional content to show for a ts:expandable control when the control is expanded.

Attributes

Attribute	Description
font	The style of the text in the hideable part of the expandable control. Valid values: bold, normal. The default is normal.
fg	The foreground color of the text in the hideable part of the expandable control, in HTML RGB format, such as '#00FF00' for green. For more information, see Using Color in User-interface Elements

Children

ts:bitmap, ts:button, ts:dropdownlist ts:edit, ts:expandable; ts:field, ts:link, ts:popupMenu, ts:popupView, ts:demandPage, ts:sync; html:br, html:table; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained in

ts:expandable; (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

A2

ts:hideable

`<ts:hideable font="normal|bold" fg="#000000">...</ts:hideable>`

The additional content to show for a ts:expandable control when the control is expanded.

Attributes

Attribute	Description
font	The style of the text in the hideable part of the expandable control. Valid values: bold, normal. The default is normal.
fg	The foreground color of the hideable part of the expandable control, in HTML RGB format, as fg='#00FF00' for green. For more information, see Using Color in User -interface Elements.

Children

ts:bitmap, ts:button, ts:drop pdownList ts:edit, ts:expandable, ts:field, ts:link, ts:popupMenu, ts:popupView, ts:demandPage, ts:sync; html:br, html:table; (xsl:apply-temp lates, xsl:call-templ ate, xsl:choose, xsl:if, xsl:for-each).

Contained in

ts:expandable; (xsl:template, xsl:when, xsl:otherwis exsl:if, xsl:for-each).

ts:edit

`<ts:edit select="xpath" xsi:type="namechars" minWidth="int" font="normal|bold" fg="#000000" focalPoint="yes|no">...</ts:edit>`

An editable text field, indicated by a dotted underscore. A ts:edit field grows to fit the contained text, unlike a text box in a form in an HTML page.

Attributes

Attribute	Description
select	An XPath expression. Required. Must evaluate to a nodeset containing exactly one node, either within the data store or a temporary tree. That node must have either no children or one text node child. The current value of the edit field is the string value of that node. The text entered in the edit field becomes the new string value for the selected node.
xsi:type	The data type the user is allowed to enter into the editable field. Can be a supported XSD "built-in type" or an XML "user-defined data type" which you define in xsd:schema section at the top of the file. Validates data entered in the field when user enters data then clicks away. See Validating User Input Using Data Types.
minWidth	The minimum width to be displayed, in pixels. The default is 5.
font	The style of the text in the editable text field. Valid values: bold, normal. The default is normal.
	If this attribute is unspecified, the style is inherited from the closest ancestor with a font attribute.
fg	The foreground color of the edit field, in HTML RGB format, such as fg='#00FF00' for green. For more information, see Using Color in User-interface Elements.
focalPoint	Whether to put the edit-focus and cursor on this edit field when the user goes to a view that has a moveFocus="first" attribute or opens an expandable that has a moveFocus="first" attribute. Valid values: yes, no. Default is "no". See Setting the Focus on a UI Element.

Children

ts:onChange, ts:onClick; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained in

ts:hideable; html:body, html:td; (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

A3

ts:appendChild

`<ts:appendChild    select="xpath">...</ts:appendChild>`

Edits your data. Determines where you are in the data tree, and appends a child to that node in the tree. If there are already children elements, adds the new node as the last child. You can add multiple children.

See also Allowed Scope of appendChild Operations.

Attributes

Attribute	Description
select	An XPath expression that indicates the location in the data tree in which to insert the resulting XML branch.

Children

Data-tree elements; xsl:attribute; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Claimed in

ts:onChange, ts:onClick; (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

Example 1

```
<ts:field font='bold'>Notes:</ts:field>
<xsl:variable name='newNote'/>
<ts:edit variable='newNote' minWidth='600'/>
<ts:button label='Close'>
    <ts:onClick>
        <ts:appendChild select='/sample:contactDB/sample:results'>
            <sample:workorderlabor>
                <sample:billablehours>0.5</sample:billablehours>
                <sample:billingrate>4</sample:billingrate>
                <sample:workorderid><xsl:value-of select='$workorderid'/></sample:workorderid>
                <sample:comment><xsl:value-of select='$newNote'/></sample:comment>
            </sample:workorderlabor>
        </ts:appendChild>
        <ts:newView href='scheduleView'/>
    </ts:onClick>
</ts:button>
```

Example 2

```
<ts:button label='done'>
    <ts:onClick>
        <ts:appendChild select='/myCom:contactDB/myCom:results'>
            <myCom:ctc myContacts='1'>
                <myCom:fnm>
                    <xsl:value-of select='$newFName'/>
                </myCom:fnm>
                <myCom:lnm>
                    <xsl:value-of select='$newLName'/>
                </myCom:lnm>
            </myCom:ctc>
        </ts:appendChild>
        <ts:newView href='main'/>
    </ts:onClick>
</ts:button>
```

A4

ts:setAttr

`<ts:setAttr name="namechars" select="xpath">string</ts:setAttr>`

Programatically sets an attribute in the d ata tree The tring that is a child of this element is the value to s et the attribut

Attributes

Attribute	Description
name	The name of the attribute to change in the data tree. Enables setting the value of an attribute which doesn't exist yet.
select	An XPath expression that selects an element in the data tree or in a temporary tree defined by an editable variable.

Children

Text nodes; xsl:value-of; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained In

ts:onChange, ts:onClick, (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

Remarks

The `<ts:setAttr>` element is provided in addition to `<ts:setText>` In order to provide a way to set the value of an attribute which doesn't exist yet.

The following sets the **statusId** attribute if it already exists:
```
<ts:setAttr select='vel:expense/@statusId'>
```

If the statusId attribute doesn't exist yet in the data tree, the select expression would evaluate to an empty nodeset.

The following sets the **statusId** attribute even if it doesn't exist yet in the data tree:
```
<ts:setAttr select='vel:expense' name='statusId'>
```

ts:sync

`<ts:sync txnPriority="int" txnBandwidth="float" select="xpath" progressive="true|false"/>`

Forces a synchronization with the server to occur. Attributes restrict in various ways the set of edits that take part in the synchronization. For example, only the edits from the views or nodes of the data tree which match the specified bandwidth and priority are sent during synchronization. (For exceptions to this rule, see "Transactional Dependencies" in Specifying Priority and Bandwidth of a Sync Event.)

You can provide multiple Synchronize buttons or menuitems that trigger different kinds of synchronization.

A related attribute is the **txnResolution** attribute on ts:view or xsd:schema. For more information, see Controlling Which Data Is Synchronized.

Attributes

Attribute	Description
txnPriority	Controls which data is sent during synchronization, based on the priority that is assigned on `<ts:view>` elements, or assigned to node-sets in the data tree as specified in an `<xs:schema>` element. Valid values: integers 1-5, with 1 being the highest priority. The default is 3.
txnBandwidth	Controls which data is sent during synchronization, based on the bandwidth value on `<ts:view>` elements, or for node-sets in the data tree as specified in an `<xs:schema>` element. Valid values: floating point values from 0.0 to large numbers (kilobits/s), -1, or "detect". The default value is -1 (meaning no bandwidth specification). The value "detect" determines the current connection speed and only sends transactions which have overall bandwidth values <= the current connection speed.
select	An XPath expression that defines the scope of the data to be sent during synchronization. Only edits that occurred within node-trees rooted at the nodes that match the expression are included in the synchronization. When this attribute is specified, the txnPriority and txnBandwidth attributes are ignored.
progressive	Causes a progressive sync, which first synchronizes priority 1 client edits, then priority 2, and so on. Valid values: true, false. If the txnPriority attribute is set, this is used as the upper bound of the progressive sync (the default is 5, which is the lowest-priority value allowable for a transaction). If the txnBandwidth attribute is set, then no transactions with a higher txnBandwidth setting is sent during the progressive sync.

A5

ts:onClick

`<ts:onClick>...</ts:onClick>`

Defines what happens when the user clicks the visible control that contains the onClick element. When the control is clicked, the element inside the onClick element is instantiated. For example, the **ts:onClick** element can be used to load a new view, which presents an entire new screen or the content of a frame, or redraws the current screen. **ts:onClick** can be used in a variety of other ways as well.

Attributes

None.

Children

ts:abortTxn, ts:appendChild, ts:back, ts:closePopupView, ts:commitTxn, ts:delete, ts:deleteTxn, ts:demandPage, ts:exportUserDatabase, ts:forward, ts:newView, ts:playSound, ts:popupMenu, ts:popupView, ts:setAttr, ts:setText, ts:sync; xsl:variable; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained In

ts:bitmap, ts:button, ts:dropdownList, ts:edit, ts:expandable, ts:field, ts:link, ts:menuItem, ts:visible; html:body, html:td; (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

ts:onChange

`<ts:onChang >...</ts:onChange>`

Defines actions to take when a UI control is used to change to data. The descendants of this element are evaluated every time a new value is copied from the edit control to the document being edited (the data tree or a temporary tree variable).

This element can be an immediate child of any edit control, including **ts:edit**, **ts:dropdownList**, and **ts:spin**.

Attributes

None.

Children

ts:abortTxn, ts:appendChild, ts:back, ts:closePopupView, ts:commitTxn, ts:delete, ts:deleteTxn, ts:demandPage, ts:forward, ts:newView, ts:playSound, ts:popupMenu, ts:popupView, ts:setAttr, ts:setText, ts:sync; xsl:variable; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained In

ts:dropdownList, ts:edit, ts:spin; (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

ts:dropdownList

`<ts:dropdownList select="`*xpath*`" match="`*xpath*`" use="`*xpath*`" show="`*xpath*`" defaultShow="`*string*`" font="normal|bold"`
`fg="#000000">`...`</ts:dropdownList>`

A user-interface control that opens to display a list of items to pick from. Used to edit the data tree or a temporary tree variable.

The popup displays its current value alongside a popup icon, shown as a triangular down-arrow. When the user clicks on the icon or the displayed value, the list appears and the user must make a choice from that list. When the user selects an item from the list, the disappears and the selected item appears. To close the list without changing the value, the user clicks anywhere other than the list box.

Attributes

Attribute	Description
select	An XPath expression. Required. Must evaluate to a nodeset containing exactly one node, either within the main document or a temporary tree variable. That node must have either no children or one text node child. The current value of the dropdown is the string value of that selected node. The item selected from the list becomes the new string value for the selected node.
match	Determines the general list of items to include in the dropdownList.
use	Indicates what value should be stored in the data tree or in a temporary tree variable when the user selects one of the choices. Using the selected node from the match query result as the context node, the use attribute is evaluated, storing the string value of the result in the data tree.
show	Indicates what is displayed in the list. Each node that is matched is used as the context node to run the query in the show attribute. The string value of the result is what is shown for that choice.
defaultShow	Determines the string shown by default in the dropdown list even if there is no actual data-tree value that matches the defaultShow string. The defaultShow attribute has no effect on the choices presented in the dropdown list, but only on the look of the dropdown list in its unselected state.
font	The style of the text in the dropdownList. Valid values: bold, normal. The default is normal.
	If this attribute is unspecified, the style is inherited from the closest ancestor with a font attribute.
fg	The foreground color of the dropdownList, in HTML RGB format, such as fg="#00FF00" for green. For more information, see Using Color in User-interface Elements.
	If this attribute is unspecified, the style is inherited from the closest ancestor with a font attribute.

Children

ts:onChange, ts:onClick, ts:deleteTxn; xsl:sort, xsl:variable; (xsl:apply-templates, xsl:call-template, xsl:choose, xsl:if, xsl:for-each).

Contained In

ts:hideable; html:body, html:td; (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

ts:commitTxn

<ts:commitTxn/>

Commits all edits (such as ts:appendChild or ts:setAttr) in the current transaction to the edit log, and opens a new transaction. This tag may be nested through any number of template calls from the view defining the transaction. In Release 1.0, transactions are one-to-one with views, so the equivalent of **ts:commitTxn** also occurs upon leaving the view.

Edits to variables are unaffected by **ts:commitTxn** .

Together with ts:abortTxn, this element gives the application author control over the granularity of edits, and thus over what may be synchronized when.

Attributes

None.

Children

None.

Contained in

ts:onChange, ts:onClick; (xsl:template, xsl:when, xsl:otherwise, xsl:if, xsl:for-each).

ts:abortTxn

<ts:abortTxn/>

Cancels the current transaction. This element can be nested through any number of template calls from the view that defines the transaction. The transaction is kept open for further editing. All edits, such as **ts:appendChild** or ts:setAttr are canceled.

Edits to variables are unaffected by <ts:abortTxn>.

Rolls back changes to local variables, so it is possible, for example, to cancel all the changes to editable fields in a view.

Together with ts:commitTxn and ts:sync, this element gives the application author control over the granularity of edits, and thus over what may be synchronized when.

For more information, see Enabling Cancel without Validating Fields.

Attributes

None.

Children

None.

Contained in

ts:onChange, ts:onClick

<!-- Appendix A -->

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<!-- Appendix A -->
<!-- Sample schema illustrating part of a sample
     embodiment of the invention. -->
<schema xmlns = "http://www.w3.org/2000/10/XMLSchema"
        targetNamespace = "http://www.thinkshare.com/ts"
        xmlns:xsl = "http://www.w3.org/1999/XSLT"
        xmlns:ts = "http://www.thinkshare.com/ts">
    <import namespace = "urn:xa:tmp:xsl"/>
    <element name = "viewRef">
        <complexType>
            <attribute name = "viewName" type = "string"/>
        </complexType>
    </element>
    <element name = "opportunity">
        <complexType>
            <choice>
                <element ref = "view"/>
                <element ref = "viewRef"/>
            </choice>
        </complexType>
    </element>
    <element name = "view">
        <complexType>
            <sequence/>
            <attribute name = "name" type = "string"/>
        </complexType>
    </element>
    <element name = "protected">
        <complexType>
            <sequence/>
        </complexType>
    </element>
    <element name = "private">
        <complexType>
            <sequence/>
        </complexType>
    </element>
    <element name = "public">
        <complexType>
            <sequence/>
        </complexType>
    </element>
    <element name = "devicemap">
        <complexType>
            <sequence/>
        </complexType>
    </element>
    <element name = "attribute">
        <complexType>
            <choice minOccurs = "0" maxOccurs = "unbounded">
                <element ref = "accessRule"/>
                <element ref = "label"/>
                <element ref = "changeScope"/>
                <element ref = "xsl:template"/>
            </choice>
        </complexType>
```

A9-a

```xml
    </element>
    <element name = "mayNotChange" type = "string"/>
    <element name = "mayChange" type = "string"/>
    <element name = "changeScope">
        <complexType>
            <sequence>
                <element ref = "mayChange" minOccurs = "0"/>
                <element ref = "mayNotChange" minOccurs = "0"/>
            </sequence>
        </complexType>
    </element>
    <element name = "long" type = "string"/>
    <element name = "medium" type = "string"/>
    <element name = "short" type = "string"/>
    <element name = "label">
        <complexType>
            <sequence>
                <element ref = "short" minOccurs = "0"/>
                <element ref = "medium" minOccurs = "0"/>
                <element ref = "long" minOccurs = "0"/>
            </sequence>
        </complexType>
    </element>
    <element name = "agent">
        <complexType>
            <sequence/>
        </complexType>
    </element>
    <element name = "action">
        <complexType>
            <sequence/>
        </complexType>
    </element>
    <element name = "accessRule">
        <complexType>
            <sequence>
                <element ref = "action"/>
                <element ref = "agent"/>
            </sequence>
        </complexType>
    </element>
    <element name = "inherit" type = "string"/>
    <element name = "element">
        <complexType>
            <choice minOccurs = "0" maxOccurs = "unbounded">
                <element ref = "inherit"/>
                <element ref = "accessRule"/>
                <element ref = "label"/>
                <element ref = "changeScope"/>
                <element ref = "opportunity"/>
                <element ref = "xsl:template"/>
                <element ref = "attribute"/>
            </choice>
        </complexType>
    </element>
    <element name = "method">
        <complexType>
```

A9-6

```
                    <sequence/>
        </complexType>
</element>
<element name = "dataSourceRef">
        <complexType>
                <attribute name = "datasourceName" type = "string"/>
                <attribute name = "serviceName" type = "string"/>
        </complexType>
</element>
<element name = "initialValue">
        <complexType>
                <sequence/>
        </complexType>
</element>
<element name = "schemaRef">
        <complexType>
                <simpleContent>
                        <extension base = "string">
                                <attribute name = "publicId" type = "string"/>
                                <attribute name = "elementName" type = "string"/>
                        </extension>
                </simpleContent>
        </complexType>
</element>
<element name = "storeDecl">
        <complexType>
                <sequence>
                        <element ref = "schemaRef"/>
                        <element ref = "initialValue"/>
                </sequence>
                <attribute name = "name" type = "string"/>
        </complexType>
</element>
<element name = "storageDecl">
        <complexType>
                <choice minOccurs = "0" maxOccurs = "unbounded">
                        <element ref = "storeDecl"/>
                        <element ref = "dataSourceRef"/>
                </choice>
                <attribute name = "name" type = "string"/>
        </complexType>
</element>
<element name = "import">
        <complexType>
                <sequence/>
                <attribute name = "localAlias" type = "string"/>
                <attribute name = "publicId" type = "uriReference"/>
        </complexType>
</element>
<element name = "servo">
        <complexType>
                <choice minOccurs = "0" maxOccurs = "unbounded">
                        <element ref = "import"/>
                        <element ref = "storageDecl"/>
                        <element ref = "view"/>
                        <element name = "xsl:template">
                                <complexType/>
```

A/0-a

```
                </element>
                <element ref = "method"/>
                <element ref = "element"/>
                <element ref = "devicemap"/>
                <element ref = "public"/>
                <element ref = "private"/>
                <element ref = "protected"/>
            </choice>
            <attribute name = "publicId" type = "uriReference"/>
            <attribute name = "localAlias" type = "string"/>
        </complexType>
    </element>
</schema>
```

A10-b

Appendix B

```
<ts:servo xmlns:wh="http://thinkshare.com/registrar/wh/schema/initial/1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xmlns:ts="urn:thinkshare.com:ts"
    name="Office Assets"
    pid="http://thinkshare.com/registrar/wh/app/init/1">

    <xsl:key name="objByLast" match='//wh:obj' use='@wh:last'/>
    <xsl:key name="objByBarId" match='//wh:obj' use='@wh:barId'/>

    <xsd:schema targetNamespace="http://thinkshare.com/registrar/wh/schema/initial/1">
        <xsd:element name = "objects">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref = "obj" minOccurs = "0" maxOccurs = "unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name = "obj">
            <ts:opportunity>
                <viewRef viewName="opObj"/>
            </ts:opportunity>
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref = "desc"/>
                </xsd:sequence>
                <xsd:attribute name = "last" type = "xsd:string"/>
                <xsd:attribute name = "barId" type = "xsd:string"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name = "desc" type = "xsd:string"/>
    </xsd:schema>

    <ts:storageDecl>
        <ts:storeDecl name="default">
            <ts:schemaRef publicId="http://thinkshare.com/registrar/wh/schema/initial/1"
                element="objects"/>
        </ts:storeDecl>
    </ts:storageDecl>

    <!-- Main View -->
    <ts:view name="main" scrolling="no">
        <xsl:call-template name="rootedObjects">
            <xsl:with-param name="root" select='key("objByLast", "0")'/>
        </xsl:call-template>
    </ts:view>

    <ts:view name="opObj" scrolling="no">
        <xsl:call-template name="rootedObjects">
            <xsl:with-param name="root" select="."/>
        </xsl:call-template>
    </ts:view>

    <xsl:template name="objBar">
        <html:body>
            <xsl:variable name="searchBarId" ts:editable="yes"><xsl:text/></xsl:variable>
            <xsl:variable name="searchDesc" ts:editable="yes"><xsl:text/></xsl:variable>
            <xsl:variable name="searchBarIdError" ts:editable="yes"><xsl:text/></xsl:variable>
            <xsl:variable name="searchDescError" ts:editable="yes"><xsl:text/></xsl:variable>

            <!-- find by barcode -->
            <ts:field>Find object using barcode</ts:field>
            <ts:edit fg="{$objIdColor}" select="$searchBarId"/>
            <ts:button label="Find">
                <ts:onClick>
                    <xsl:variable name='normedBarId' select='normalize-space($searchBarId)'/>
```

```xml
            <xsl:choose>
                <xsl:when test='$normedBarId!=""'>
                    <xsl:variable name='foundBarId' select='key("objByBarId", $searchBarId)'/>
                    <xsl:choose>
                        <xsl:when test="$foundBarId">
                            <ts:setText select="$searchBarIdError">
                                <xsl:text></xsl:text>
                            </ts:setText>
                            <ts:newView href="rootedObjects" select=".">
                                <xsl:with-param name="root" select="$foundBarId"/>
                            </ts:newView>
                        </xsl:when>
                        <xsl:otherwise>
                            <ts:setText select="$searchBarIdError">
                                <xsl:text>Not found</xsl:text>
                            </ts:setText>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
                <xsl:otherwise>
                    <ts:setText select='$searchBarIdError'>
                        <xsl:text>Need non-space characters</xsl:text>
                    </ts:setText>
                </xsl:otherwise>
            </xsl:choose>
        </ts:onClick>
    </ts:button>
    <ts:field fg="{$errorColor}">
        <xsl:value-of select="$searchBarIdError"/>
    </ts:field>
    <html:br/>

    <!-- find by description substring -->
    <ts:field>Find using substring</ts:field>
    <ts:edit fg="{$descColor}" select="$searchDesc"/>
    <ts:button label="Find">
        <ts:onClick>
            <xsl:choose>
                <xsl:when test='string-length($searchDesc)>2'>
                    <xsl:variable name="foundDesc" select="/wh:objects/wh:obj[contains(wh:desc,
$searchDesc)]"/>
                    <xsl:choose>
                        <xsl:when test="$foundDesc">
                            <ts:setText select="$searchDescError">
                                <xsl:text></xsl:text>
                            </ts:setText>
                            <ts:newView href="rootedObjects" select=".">
                                <xsl:with-param name="root" select="$foundDesc"/>
                            </ts:newView>
                        </xsl:when>
                        <xsl:otherwise>
                            <ts:setText select="$searchDescError">
                                <xsl:text>Not found</xsl:text>
                            </ts:setText>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:when>
                <xsl:otherwise>
                    <ts:setText select="$searchDescError">
                        <xsl:text>More characters</xsl:text>
                    </ts:setText>
                </xsl:otherwise>
            </xsl:choose>
        </ts:onClick>
    </ts:button>
    <ts:field fg="{$errorColor}">
        <xsl:value-of select="$searchDescError"/>
    </ts:field>
</html:body>
```

```
    </xsl:template>
        <xsl:param name="root"/>
        <xsl:param name="offerParent" select='"no"'/>
        <html:frameset rows="51, *">
            <html:frame border="bottom" src="objBar" scrolling="yes"/>
            <html:frame src="objTree">
                <xsl:with-param name="root" select="$root"/>
                <xsl:with-param name="offerParent" select="$offerParent"/>
            </html:frame>
        </html:frameset>
    </ts:view>

</ts:servo>
```